# On curves and computation with shapes

## Conference or Workshop Item

# ON CURVES AND COMPUTATION WITH SHAPES

IESTYN JOWERS, MIQUEL PRATS, CHRIS EARL, STEVE GARNER
*The Open University*

**Abstract.** Few computer implementations of shape grammars have been written that are applicable to curved shapes. The reason for this is that underlying representations of curves do not lend themselves to providing a unique description of curved shapes. Such a representation is essential for shape algorithms. This paper explains the problems posed in representing curves for shape grammar implementation. The paper demonstrates how the fundamental structures for implementing shape grammars for straight lines can be extended to shapes with curved segments.

## 1. Introduction

Shape grammars provide an ideal foundation for new computer based generative tools that will allow designers to synthesize designs and explore design spaces. Because of their ability to recognize and operate on emergent shapes they can easily produce interesting and unexpected designs from a small set of simple shape rules. They have been implemented repeatedly in architecture, most commonly to capture the style of a specific architect for example the Malagueira grammar (Duarte, 2000), but they have rarely been applied in product design. This is mainly due to the fact that until recently shape grammar programs that take advantage of the emergent properties of shapes were, for technical reasons, written only to work with shapes composed solely of straight lines. Investigations into the computer implementation of a curved shape grammar have been carried out by Chau (2002) and McCormack and Cagan (2003), who have implemented examples of shape grammar systems for parametric curves. These are important developments, since in order for shape grammars to be useful in product design it is essential that they work with curves so as to produce the organic forms of which so many products consist.

This paper outlines the development of new fundamental structures for curves based on the maximal lines method of Krishnamurti (1981) that will allow implementation of parametric shape grammars. First, a brief review of

1. Author of contact: i.r.jowers@open.ac.uk

computer implementations of shape grammars is presented in which the issues regarding implementation are introduced. This is followed by a discussion regarding the issues of curve representation. Finally, the implementation of a curved shape grammar is considered.

## 2. Shape Grammars

A shape grammar consists of an initial shape and a set of shape rules of the form $\alpha \rightarrow \beta$, where $\alpha$ and $\beta$ are both shapes. A shape rule is applicable to a shape $\gamma$ if there is a transformation $\tau$ such that $\tau(\alpha)$ is a subshape of $\gamma$. In such a case the occurrence of $\tau(\alpha)$ in $\gamma$ is replaced with $\tau(\beta)$ (Stiny, 1980). Shape rules are applied repeatedly to the initial shape to create a sequence of designs in a design space. Often applying a small set of simple rules to a shape can result in interesting and unexpected results (Stiny, 1994). For example, in Figure 1 a shape grammar *SG1* is given. The grammar consists of a single rule *SR1* that translates a curved shape $\alpha$ along its own length, Figure 1a), and an initial shape *IS1*, Figure 1b).

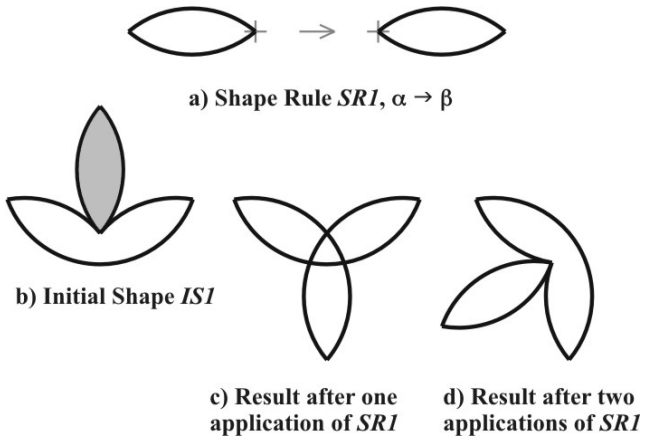


Figure 1: *Shape Grammar SG1*

*IS1* contains one instance of the shape $\alpha$, highlighted in grey. Application of *SR1* to *IS1* can result in two different designs; one example is given in Figure 1c). The new shape contains three instances of $\alpha$; two new instances have emerged. Application of *SR1* to this shape can result in six different designs; one example is given in Figure 1d). This design is the initial shape rotated through an anti-clockwise angle of $120^{o}$ but the rule that caused the rotation is itself merely a translation. This unexpected result was brought about by the shape grammar's ability to recognize and operate on the emergent shapes in a design.

## 3. Computer Implementation of Shape Grammars

Since their conception efforts have been made to write computer programs that automate the implementation of shape grammars. Such implementations are necessary if shape grammars are to be part of an automated generative system. These programs generally fall into three categories: application specific implementations, spatial relation implementations, and shape grammar interpreters.

### 3.1 APPLICATION SPECIFIC IMPLEMENTATIONS

Application specific implementations explore particular design spaces by predetermined sets of shape rules. In general, they are produced as a demonstration of the potential for shape grammars to be used in different modes of design. For example, the coffeemaker grammar (Agarwal and Cagan, 1998) is a demonstration of a shape grammar implementation that utilizes function labels to produce a large class of product designs. Similarly, the motorcycle grammar (Pugliese and Cagan, 2002) is a demonstration of a shape grammar implementation that captures a specific brand identity, and produces a class of designs within that brand, and the clock grammar (Starling and Shea, 2002) is a demonstration of a parallel shape grammar implementation that generates mechanical systems by considering both their functional and structural requirements.

However, it is common for implementations of this type to be driven by labels, rather than the emergent properties of shapes themselves. This results in shape grammars that are implemented in a restrictive manner.

The truss grammar (Shea and Cagan, 1998) is an example of an application specific implementation that does not depend on labels to drive the grammar. However, the shapes considered by the grammar are represented merely by points and their connecting lines, and emergent properties of the shapes are not considered.

### 3.2 SPATIAL RELATION IMPLEMENTATIONS

Spatial relation implementations allow the user to define shape rules based on the spatial relations between shapes, which can be used in the exploration of a wide variety of design spaces. Generally, these programs are based on a highly restricted type of shape grammar called basic shape grammars (Knight, 1999). Basic shape grammars are deterministic and are composed solely of addition rules, i.e., rules that add a shape. The rules are linearly ordered and each applies under one similarity transformation to the shape added by the previous rule. As a result, the emergent properties of shapes do not drive grammars of this type. For example, Shaper 2D (McGill, 2001) is a two-dimensional spatial relation implementation that allows the exploration

of design spaces derived from the spatial relations between two given shapes: squares, rectangles or triangles. Similarly, 3DShaper (Wang and Duarte, 2002) is a three-dimensional spatial relation implementation that allows the exploration of design spaces derived from the spatial relations between two orthogonal shapes such as pillars, oblongs, and cubes. Both of these implementations are based on basic shape grammars and the emergent properties of the shapes do not drive the grammars.

### 3.3 SHAPE GRAMMAR INTERPRETERS

Shape grammar interpreters are programs that aid in the generation of shapes from shape grammars. They allow application of rules that recognize and operate on emergent shapes, as in the example given in Figure 1. As a result, they utilize the full generative powers of shape grammars. The earliest shape grammar interpreter to be implemented was by Gips (1975), but the interpreter avoided the issue of detecting emergent shapes. The first shape grammar interpreter that was able to detect emergent shapes was implemented by Krishnamurti (1982), and was based on shape algorithms that use a maximal line description to detect emergent shapes (Krishnamurti, 1980). Since then, two-dimensional interpreter programs have been written by Krishnamurti and Giraud (1986), Chase (1989) and Tapia (1999), all of which are based on the shape algorithms of Krishnamurti and as such are non-parametric interpreters that work with shape grammars consisting solely of straight lines.

However, McCormack and Cagan (2003) have developed a parametric interpreter that implements shape grammars that consist of curved shapes. The interpreter works by determining a straight-line equivalent of a curved shape, which is then used, with reference to the original curved shape, to detect emergent shapes. It has been used to implement the Buick brand shape grammar (McCormack, Cagan, *et al.,* 2004).

Of the three types of implementation discussed above, shape grammar interpreters remain closest to the philosophy of shape grammars in that they allow the emergent properties of a shape to drive a grammar. Three operations are necessary in order to implement any shape grammar. These are shape recognition, shape difference and shape union, and the underlying difficulty with writing shape grammar interpreters is the question of how to represent shapes so that these operations can be applied.

#### *3.3.1 The Method of Maximal Lines*

The shape algorithms developed by Krishnamurti (1980) are the basis for most straight line shape grammar interpreters. The algorithms use a maximal line representation of shapes, where a line segment in a shape is defined to be maximal when no other line segments in that shape contains it. Every

maximal line is considered to be a finite segment of an infinite line, and can therefore be associated with a line descriptor ψ that defines the position and orientation of the infinite line on which it lies. For example, ψ can be given by the ordered pair ψ = (μ, ν), where μ is the slope of the line and ν is the y-intercept if the line is non-vertical and the x-intercept if the line is vertical. Any collinear maximal lines lie on the same infinite line and have the same line descriptor, whereas non-collinear lines do not. Every shape that is composed of straight lines is broken up into maximal collinear lines, where each set lies on an infinite line and is given a line descriptor, as shown in Figure 2.


Figure 2: *Decomposition of a shape into sets of collinear maximal lines*

Maximal lines are positioned on an infinite line by their end points, which are referred to as the tail and head of the line. These end points can be ordered so that tail < head. Here < is an order relation and is defined as follows: Let $a$, $a = (a_1, a_2, \ldots, a_n)$, and $b$, $b = (b_1, b_2, \ldots, b_n)$ be two n-tuples of numbers, then a < b whenever there is a $k$ such that $a_k < b_k$, and $a_j = b_j$ for all $j < k$. In this way each set of collinear lines is reduced to an ordered list of end points. Similarly the sets are arranged so that their line descriptors form a linearly ordered list, and as a result every unique shape is given a unique numerical representation. Shape operations can then be applied by comparing lists of numerical data.

*3.3.2 A Shape Grammar Interpreter for Curved Shapes*

In a recent paper, McCormack and Cagan (2003) describe a Parametric Shape Grammar Interpreter that applies shape rules to curved shapes. The interpreter determines a straight-line shape equivalent of a curved shape, referred to as a distinct shape, by connecting distinct points of the shape with straight lines for example in Figure 3 a curved shape α is shown with its distinct shape, which is a square.

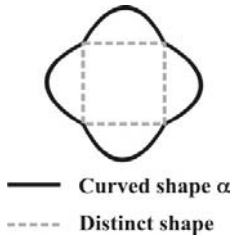


Figure 3: *A curved shape α and its distinct shape*

Distinct points are determined by the user and can include points such as endpoints and points of intersection. The curved shape is reduced to a line shape, the line segments of which correspond to the curves of the original shape. For curved shapes with obvious distinct points, this method provides a unique representation of the shape i.e., a distinct line shape with the properties of the original curves acting as labels on the lines. Shape operations can then be applied to the distinct shape, with reference to the original curved shape.

However, for shapes without any obvious distinct points, for example a closed continuous curve, a unique representation of the shape is not provided. Instead, an infinite number of distinct shapes can be found to represent the curved shape. The shape is not defined uniquely and the operations necessary to implement a shape grammar cannot be applied. If for such curves a procedure was imposed that would choose specific points on the curve to be distinct points, for example in Figure 4 a circle is reduced to a distinct shape that is a square, then properties of the original shape, for example in this case the infinite symmetry of the circle, are lost, and such a loss would occur no matter how the distinct shape is chosen. This results in a restricted application of shape grammars.

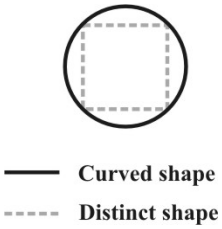


Figure 4: *A circle and its possible distinct shape*

## 4. Computation with Curves

In order to computationally implement a shape grammar, shapes in the language of the grammar need be described uniquely. For computational efficiency, it is useful for the shapes to be given a unique canonical form that

can be described by a finite ordered list of rational numbers. Performing shape operations is then reduced to comparing lists of numerical data.

It was shown above that representing a curved shape in terms of lines results in restricted implementation of shape grammars. In order to implement shape grammars to curved shapes unrestrictedly it is necessary to describe the shape in terms of the curves of which it is composed. If we are to follow the logic of the method of maximal line, we can obtain a unique canonical form of a curved shape by breaking it up into curved primary segments e.g., maximal curve segments, which lie on curved carriers e.g., infinite curves.

### 4.1 HOW TO DESCRIBE A SHAPE

All shapes can be broken into primary segments, and as a result can be described by those primary segments. For example a square can be broken into four lines, and four lines can describe a square. But, a shape can be broken in an infinite number of ways, for example a square can also be broken into eight lines, two L's or two C's etc, Figure 5.

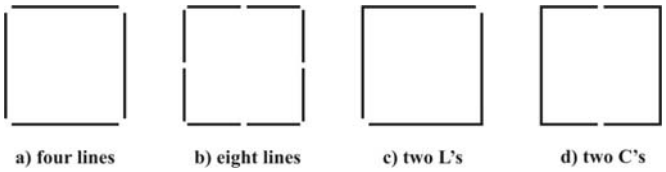


Figure 5: *A square broken into primary segments*

A square can be broken into eight lines, two L's, or two C's in an infinite number of ways hence the square cannot be uniquely described in terms of those primary segments. On the other hand, a square can be broken into four lines in only one way and hence can be uniquely described by them. These four lines are the maximal lines of the square since no other line segments in the square contain them. Similarly, any shape composed of straight lines can be uniquely described by its maximal lines, and any shape composed of curved lines can be uniquely described by its maximal curves.

In the method of maximal lines, line segments are uniquely described in terms of the line carriers on which they lie. Similarly maximal curve segments can be uniquely defined in terms of the curve carriers on which they lie. These maximal segments are described on three levels:

a) The type of carrier on which the segment lies
b) The position and orientation of the carrier on which the segment lies
c) The location of the segment on its carrier

*a) The type of carrier on which the segment lies*

In the method of maximal lines, the first level of description is not required since all the maximal segments in line shapes lie on the same type of carriers i.e., line carriers. A curved shape however may be composed of maximal segments that lie on a variety of different types of carriers. As a result, a distinction needs to be made between these different types, for example quadratic polynomial curves need to distinguished from trigonometric curves, logarithmic curves need to be distinguished from cubic monomial curves, etc.

*b) The position and orientation of the carrier on which the segment lies*

In the method of maximal lines, the line carrier is defined by a line descriptor that describes the position and orientation of the line. The position and orientation of a curve carrier can similarly be described. For example, the coefficients of a polynomial curve can be used to define a curve descriptor that will describe the position and orientation of a polynomial carrier.

*c) The location of the segment on its carrier*

In the method of maximal lines, the location of a segment on a carrier is given by the endpoints of the line segment. Similarly, a primary segment on a curve carrier can also be given by the endpoints of the curve segment.

Any curve segment that is described on these three levels has a unique description, as does a shape composed of such curve segments.

## 4.2 HOW TO DESCRIBE A CURVE

### *4.2.1 A Curve Segment described in terms of its Carrier*

In the method of maximal lines a finite segment i.e., a line segment is described in terms of the carrier on which it lies i.e., an infinite line. The segment's position on the carrier is given in terms of its endpoints. Similarly a curve segment can be described in terms of the carrier on which it lies. For example, pieces of circular arc can be described in terms of the circles on which they lie. The arcs can be defined to be maximal so that if any two arcs that lie on the same carrier have points in common they can be considered to compose a single arc. Consider again the shape grammar *SG1*, Figure 6, which is implemented using maximal arcs to describe the shapes. The carriers on which the arcs lie are shown.

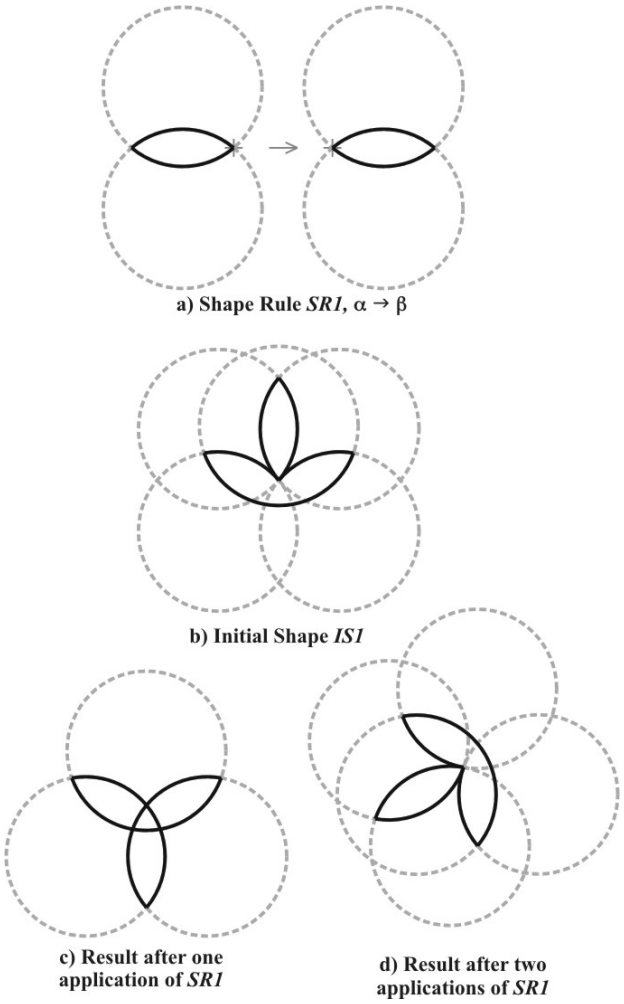


Figure 6: *Shape grammar SG1 implemented using maximal circular arcs*

The grammar is implemented successfully. *SR1* translates the curved shape α along its own length, Figure 6a). *IS1* contains one instance of α, Figure 6b). A single application of SR1 results in Figure 6c). Here, there are two instances where two separate arcs, which are both defined on the same carrier, are combined to form a single curve, as a result two more instances of α have emerged. A second application of *SR1* results in Figure 6d).

Theoretically, a shape grammar interpreter could be based on curve segments described in terms of their carrier. The user could describe shapes using curve segments that lie on a predetermined class of carriers, for example conic curves, which could then be used to describe the language of

a shape grammar. However, such a system could restrict the freedom of the user, who may wish to use more freeform curves in a shape grammar.

*4.2.2 A Curve Carrier described in terms of a Segment*

Alternatively, in the method of maximal lines, a finite segment describes the carrier on which it lies i.e., a line segment defined by two end points uniquely defines the infinite line on which it lies. Similarly a curve segment can be defined by a finite number of points, and can define the infinite curve on which it lies. For example a Bézier curve is a curve segment defined by a finite number of control points. It is defined on an interval [0, 1], but lies on an infinite parametric curve which is defined by the equation of the Bézier curve. Two Bézier curves $B_1(t_1) = [f_1(t_1), g_1(t_1)]$, $0 \leq t_1 \leq 1$ and $B_2(t_2) = [f_2(t_2), g_2(t_2)]$, $0 \leq t_2 \leq 1$ lie on the same carrier if there exists a linear transformation $\tau$ of $t_2$, such that $B_1(t_1) = B_2(\tau(t_2))$. The curves can be defined to be maximal so that if two such curves have any points in common they can be combined to form a single curve segment $B_3(t_3) = [f_3(t_3), g_3(t_3)]$, $0 \leq t_3 \leq 1$, which also lies on the same carrier. Consider again the shape grammar *SG1*, Figure 7, which is now implemented using maximal cubic Bézier curves to describe the shapes. The control polygons of the Bézier curves are shown.

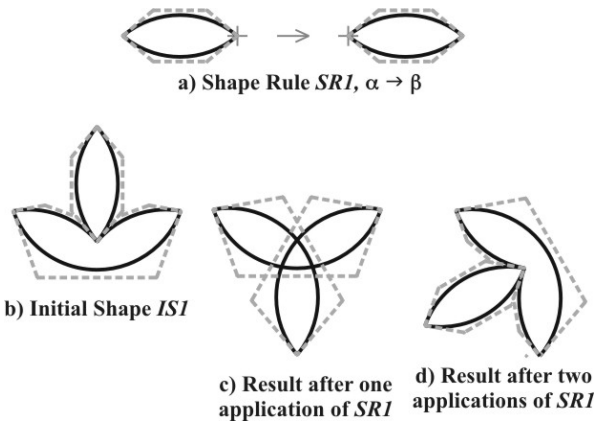


Figure 7: *Shape grammar SG1 implemented using maximal cubic Bézier curves*

Here again the grammar is implemented successfully. *SR1* translates the curved shape α along its own length, Figure 7a). *IS1* contains one instance of α, Figure 7b). A single application of *SR1* results in Figure 7c). Here, there are two instances where two separate Bézier curves, which are both defined on the same carrier, are combined to form a single curve, as a result two more instances of α have emerged. A second application of *SR1* results in Figure 7d). In this example, the circular arcs in the shape are approximated

by cubic Bézier curve segments that lie on carriers of the form shown in Figure 8.

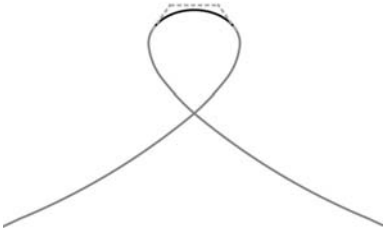

Figure 8: *Example of a curve carrier for a cubic Bézier curve*

It is important to note here that curve segments that lie on a carrier must be of the same type as the carrier itself. Hence, only curves of the same type can be added together, and the resultant curve will also be of the same type. For example, in Figure 9 *SG1* is implemented again but this time the shapes are described using a mixture of maximal cubic and quadratic Bézier curves. Again, the control polygons are shown.

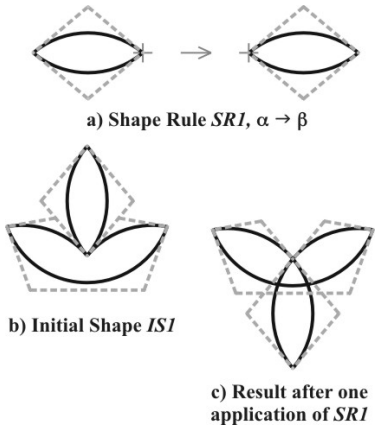


Figure 9: *Shape grammar SG1 implemented using maximal cubic and quadratic Bézier curves*

Here, the curved shape $\alpha$ in the shape rule *SR1* is described using cubic Bézier curves, Figure 9a), and the initial shape is described using a combination of both cubic and quadratic Bézier curves, Figure 9b). As before, there is an instance of $\alpha$ in the initial shape. Application of *SR1* results in Figure 9c). At first glance, it seems that two additional instances of $\alpha$ have emerged as before, but the two quadratic curves cannot be added

together to form a cubic curve therefore the moving piece remains separate from the rest of the shape, and the implementation is unsuccessful.

Allowing a curve carrier to be defined in terms of a segment would provide a very flexible theoretical base for a shape grammar interpreter if a consistent method of description were used. The user could describe shapes using freeform parametric curve segments which would in turn describe the carriers on which they lie. These shapes could then be used to describe the language of a shape grammar.

*4.2.3 Curve Carrier described Piecewise*

Another alternative is to allow a curve carrier to be described piecewise i.e., the curve carrier could be a composition of curve segments. As a result any curve that can be drawn with a pencil can be considered to be a curve carrier.

However, in order to computationally implement a shape grammar it is necessary that the shapes in the grammar be described uniquely. A result of this is that two distinct carriers cannot be allowed to share segments. If curve carriers were described piecewise, then it is feasible for two distinct carriers to share a segment, as seen in Figure 10, where two distinct curves $f(x, y)$ and $g(x, y)$ share a curve segment between the points $(x_0, y_0)$ and $(x_1, y_1)$.

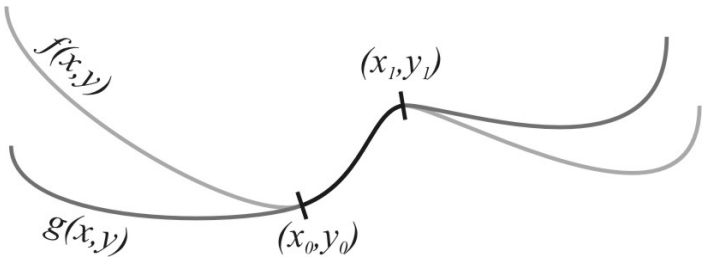


Figure 10: *Example of two distinct curves f(x, y) and g(x, y) that share a segment between* $(x_0, y_0)$ *and* $(x_1, y_1)$

If such a situation were to occur, the curve segment between the points $(x_0, y_0)$ and $(x_1, y_1)$ would not have a unique description, but could be described either in terms of the carrier $f(x, y)$ or in terms of the carrier $g(x, y)$. In order to avoid such a situation, a curve carrier cannot be allowed to be described piecewise but must instead be described by a real mathematical function. The restrictions on the mathematical functions that can be used to describe a carrier remain to be investigated.

## 4.3 SHAPE RULE APPLICATION

Computations with shapes using shape grammars can be reduced to three operations:

- Sub-shape recognition
- Shape difference
- Shape union

For shapes that are described in terms of primary segments, these operations are reduced to determining whether shapes have segments in common, for the sub-shape operation the segments are common under transformation. The operations are applied through three stages that relate to the three levels on which the primary segments are described:

1. Compare the type of carriers on which primary segments lie – if two shapes have any primary segments in common, then those primary segments must lie on the same type carrier.
2. Compare the position and orientation of the carriers on which the primary segments lie – if two shapes have any primary segments in common then those primary segments must lie on carriers with the same position and orientation, or in the case of the sub-shape operator, carriers that are congruent i.e., carriers that are the same under transformation.
3. Compare the location of the primary segments on their carriers – if two shapes have any primary segments in common then those primary segments must lie in the same location on their carriers.

For curved shapes, how these three stages are carried out depends on how the curved segments that compose the shape are described, whether implicitly, parametrically, or intrinsically.

## 5. Implementation of a Curved Shape Grammar

A simple example will now be given to illustrate how the process described above can be practically applied in order to implement a curved shape grammar. In the example, three representations of curves are considered:

1. An implicit representation – curves are described in terms of a function, $f(x, y) = 0$
2. A parametric representation – curves are described in terms of two parametric functions, $C(t) = [f(t), g(t)]$
3. An intrinsic representation – curves are described in terms of their intrinsic properties curvature, $\kappa$, and arc length, $s$, $\kappa = f(s)$

Consider the curved shape $\gamma$ in Figure 11a). It is composed of a variety of different types of curve – two circular arcs (I and III), a parabolic curve (II) and a cubic curve (IV), as shown in figure 11b)

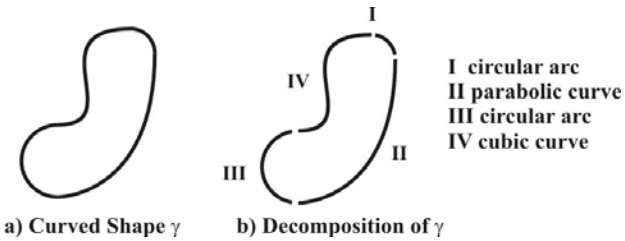


Figure 11: *A curved shape γ and it decomposition into curved segments*

Now, consider the shape rule *SR2* of the form α → β, where β is a rigid body transformation of α, Figure 12.

Figure 12: *Shape Rule* SR2

In order to determine whether *SR2* is applicable to γ it is necessary to determine whether there is an occurrence of α in γ. This is achieved by comparing the function that defines α with the components of γ.

STEP 1 - COMPARE THE TYPE OF CARRIERS

Following the procedure outlined above the first step is to compare the carriers that α and the component curves of γ lie on.

If the curves are described implicitly, e.g., $f(x, y) = 0$, then the functions that describe the curves are compared. The carrier of the cubic curve will not be of the same type as the carrier of α because a circle is described implicitly by a quadratic equation, Figure 13.

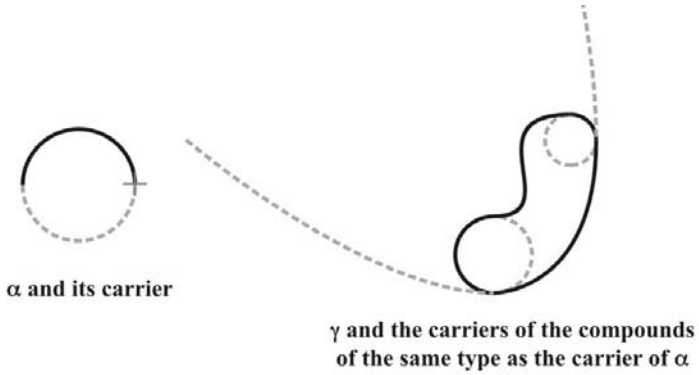


Figure 13: *Implicit comparison of the carriers of α and γ*

If the curves are described parametrically, e.g. $C(t) = [x(t), y(t)]$, then the parametric functions that describe the curves are compared. The carriers of the cubic curve and the parabolic curve will not be of the same type as the carrier of α because their parameter functions are polynomials while the parametric functions of a circle are trigonometric, Figure 14.

If the curves are described intrinsically e.g., $\kappa = f(s)$, where $s$ is the arc length along the curve, then the functions that describe the curvature of the curves are compared. The carriers of the cubic curve and the parabolic curve will not be of the same type as the carrier of α because they do not have a constant curvature whereas a circle does, Figure 14.

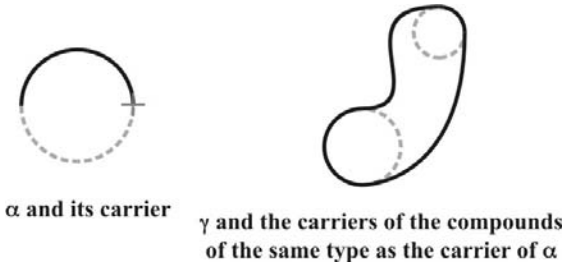


Figure 14: *Parametric and Intrinsic comparison of the carriers of α and γ*

In this step, the parametric and intrinsic representations of the curves have proved advantageous over the implicit representation because they were able to distinguish between circles and parabolas.

## STEP 2 - COMPARE THE POSITION AND ORIENTATION OF CARRIERS

After Step 1 it has been determined that two components of γ lie on the same carrier as α. The next step is to compare the position and orientation of the carriers to determine whether they are congruent. This step cannot be carried out if the curves are described intrinsically, since an intrinsic description has no information about spatial position. Instead, implicit or parametric descriptions of the curve have to be used.

If the curves are described implicitly then a transformation has to be found that maps the function that describes the carrier of α onto the functions that describe the carriers of the components of γ.

If the curves are described parametrically then a transformation has to be found that maps the parametric functions that describe the carrier of α onto the parametric functions that describe the carriers of the components of γ. This transformation has to take into account a possible change of parameter, $t \rightarrow t'$.

Because α is a circular arc, there are an infinite number of transformations that will map the carrier of α onto the carriers of the circular arcs in γ.

In this step, the implicit representation of the curves has proved to be advantageous over the parametric and intrinsic representations. Comparison of position and orientation of curves by parametric representation requires additional consideration of a change of parameter as well as a transformation of the carrier function, and comparison by intrinsic representation is not possible.

STEP 3 - COMPARE LOCATION ON CARRIERS

Following Step 2, two possible occurrences of α in γ remain. The final step in the process is to compare the location of the curve segments on their carriers.

If the curves are described implicitly, then the location of the segments on their carriers can be defined by their endpoints, $(x_0, y_0)$ and $(x_1, y_1)$. If there is a transformation that maps the carrier of α onto a carrier of a circular arc in γ, that also maps the endpoints of α onto the endpoints of the component in γ then the component is a transformation of α.

If the curves are described parametrically, then the location of the segments on their carriers can be defined by the parameter values of their endpoints, $t_0$ and $t_1$. If there is a transformation and a change of parameter, $t \rightarrow t'$, that maps the parametric functions that describe the carrier of α onto the parametric functions that describe the carriers of a component of γ and maps the parameters of the endpoints of α to the parameters of the endpoints of the component in γ, then the component is a transformation of α.

If the curves are described intrinsically, then the location of the segments on their carriers can be defined by $s_0$ and $s_1$, the arc length values of their endpoints. If there is a transformation of arc length $s \rightarrow s'$, that maps the arc length values of α to the arc length values of a component of γ, then the component is a transformation of α.

Of the two circular arcs in γ, only the semi-circle will be similarly located on its carrier as α. Therefore, one unique match of a transformation of α in γ has been found.

In this step, the implicit and intrinsic representations of the curves have proven to be advantageous over the parametric representation. Comparison of the location of curve segments on their carriers by implicit representation requires consideration of a transformation of the carrier function, and comparison by intrinsic representation requires transformation of a parameter (arc-length), whereas comparison by parametric representation requires both.

STEP 4 - SHAPE DIFFERENCE

Once an occurrence of a transformation of α has been discovered in γ, the occurrence is removed with the shape difference operator, Figure 15.

Figure 15: *Result of the shape difference operation*

STEP 5 - SHAPE UNION

The shape τ(β), where τ is the transformation under which there is an occurrence of α in γ, is then added to the shape γ - τ(α) with a shape union operator, Figure 16, and the rule has successfully been applied.

Figure 16: *Result of the shape union operation*

The three representations of curve used in this example each proved advantageous in certain steps but disadvantageous in others. As a result, a combination of representations will be utilized in future implementations.

## 6. Conclusions

In this paper, computer implementations of shape grammars have been investigated, and three types of implementation were defined – application specific implementations, spatial relation implementations, and shape grammar interpreters. Of these three types, only shape grammar interpreters were found to utilize the full generative powers of shape grammars by recognizing and operating on emergent shapes.

Fundamental structures for representing curved shapes were introduced based on the method of maximal lines. The structures allow shape computations with curves. Curved shapes are broken into maximal curved segments that lie on curved carriers. Two methods of relating the curve segments and their carriers were presented. In the first, curve segments are

described in terms of the carriers on which they lie; in the second, curve carriers are described in terms of a curve segment.

Curved shapes are now described uniquely by the unique curved segments that compose them. These curved segments are described on three levels – the type of carrier on which the segment lies; the position and orientation of the carrier on which the segment lies; and the location of the segment on its carrier.

An example of how a shape rule can be applied based on a method of maximal curves was presented, and in the example, three representations of curves were considered – implicit, parametric, and intrinsic. It was found that, in order to successfully implement a shape grammar based on this method, a combination of representations is needed.

Future work will concentrate on developing this method of maximal curves into three dimensions. A structure for 3D curves and curved surfaces is necessary if shape grammars are to become the basis of an automated generative system for product design.

## References

Agarwal, M. and Cagan, J.: 1998, A blend of different tastes: the language of coffeemakers: *Environment and Planning B-Planning & Design*, 25(2): 205-226.

Chase, S. C.: 1989, Shapes and Shape Grammars - from Mathematical-Model to Computer Implementation: *Environment and Planning B-Planning & Design*, 16(2): 215-242.

Chau, H. H.: 2002, Preserving brand identity in engineering design using a grammatical approach: *PhD Thesis*, School of Mechanical Engineering, University of Leeds.

Duarte, J. P.: 2000, A digital framework for augmenting the architect's creativity: *Digital Creativity Symposium*, Greenwich, London.

Gips, J.: 1975, *Shape Grammars and their Uses: Artificial Perception, Shape Generation and Computer Aesthetics*, Basel, Birkhauser.

Knight, T. W.: 1999, Shape grammars: six types: *Environment and Planning B-Planning & Design*, 26(1): 15-31.

Krishnamurti, R.: 1980, The arithmetic of shapes: *Environment and Planning B*, 7: 463-484.

Krishnamurti, R.: 1981, The construction of shapes: *Environment and Planning B*, 8: 5 - 40.

Krishnamurti, R.: 1982, SGI: a shape grammar interpreter: *Tech. Report*, Design Discipline, The Open University.

Krishnamurti, R. and Giraud, C.: 1986, Towards a shape editor: the implementation of a shape generation system: *Environment and Planning B: Planning and Design*, 13: 391-404.

McCormack, J. P. and Cagan, J.: 2003, Increasing the scope of implemented shape grammars: a shape grammar interpreter for curved shapes. *ASME 2003 International Design Engineering Technical Conferences and Information in Engineering Conference*, Chicago, Illinois, USA.

McCormack, J. P., Cagan, J., et al.,: 2004, Speaking the Buick language: capturing, understanding and exploring brand identity with shape grammars: *Design Studies*, 25(1): 1-29

McGill, M. C.: 2001, A visual approach for exploring computational design: *MSc Thesis*, Department of Architecture. Cambridge, Massachusetts Institute of Technology.

Pugliese, M. J. and Cagan, J.: 2002, Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar: *Research in Engineering Design-Theory Applications and Concurrent Engineering*, 13(3): 139-156.

Shea, K. and Cagan, J.: 1998, Topology design of truss structures by shape annealing: *ASME 1998 Design Engineering Technical Conferences*, Atlanta, Georgia, USA.

Starling, A. C. and Shea, K.: 2002, A clock grammar: the use of a parallel grammar in performance-based mechanical synthesis: *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Montreal, Canada.

Stiny, G.: 1980, Introduction to Shape and Shape Grammars: *Environment and Planning B-Planning & Design*, 7(3): 343-351.

Stiny, G.: 1994, Shape Rules - Closure, Continuity, and Emergence: *Environment and Planning B-Planning & Design*, 21: S49-S78.

Tapia, M.: 1999, A visual implementation of a shape grammar system, *Environment and Planning B*, 26(1): 59-74.

Wang, Y. F. and Duarte, J. P.: 2002, Automatic generation and fabrication of designs: *Automation in Construction*: 11(3): 291-302.